Smart Passenger Counter - Project Report

1. Abstract

This project implements a Smart Passenger Counter system designed to accurately track and count people entering and exiting a designated area in real-time. By leveraging computer vision techniques, specifically YOLO (You Only Look Once) for object detection and SORT (Simple Online and Realtime Tracking) for object tracking, the system provides a robust solution for automated passenger analytics. The system accounts for bi-directional flow, detecting whether a person is entering or leaving, and provides live analytics via a web interface.

2. Introduction

Automated passenger counting is critical for public transportation, retail analytics, and venue management. Traditional methods often rely on hardware sensors which can be expensive and inflexible. This project presents a software-based approach using standard video feeds.

The objective is to build a real-time system that:
- Detects humans in video streams.
- Tracks individual movements frame-by-frame.
- Counts crossings over a virtual line to determine direction (IN/OUT).
- Displays live statistics through a web API.

3. System Architecture

The system works in a pipeline of three main stages:

A. Detection (YOLOv8):
The system uses the YOLOv8 neural network (ONNX format) to detect objects in each video frame. It filters detections to retain only the 'person' class with a confidence score above a set threshold (e.g., 0.4).

B. Tracking (SORT):
To associate detections across frames, the SORT algorithm is used. It employs a Kalman Filter to predict object locations and the Hungarian Algorithm to match new detections to existing tracks based on Intersection over Union (IoU).

C. Counting Logic:
A virtual line is defined in the frame (e.g., y=360). The system monitors the centroid of each tracked person. If a person's centroid crosses the line from top to bottom, it increments the 'IN' counter. Crossing from bottom to top increments the 'OUT' counter.

4. Implementation Details

The project is implemented in Python and structured as follows:

- core/detection_tracker.py: Orchestrates the pipeline, feeding video frames to YOLO and then to SORT.
- core/passenger_counter.py: Maintains the state of each track ID (previous position) and determines if a line crossing occurred.
- core/input_reader.py: Handles video input from webcam, file, or RTSP stream in a separate thread for performance.
- server/api.py: A Flask-based web server that streams the processed video and serves a JSON API for analytics.

Dependencies include OpenCV for image processing, PyTorch for deep learning (optional if using ONNX runtime), and Flask for the web interface.

5. Conclusion

The Smart Passenger Counter successfully demonstrates the capability of modern computer vision to solve real-world problems. The system achieves real-time performance on standard hardware by optimizing the detection-tracking pipeline. Future improvements could include integrating ReID (Re-identification) to handle occlusions better and deploying the solution to edge devices for tracking on buses or trains.